Written Assignment 8

Introduction To Computer Systems (Carnegie Mellon University)

15-213: Introduction to Computer Systems
Written Assignment 8

This written assignment covers virtual memory.

Directions

Complete the questions on the following pages with single paragraph answers. These questions are not meant to be particularly long! Once you are done, submit this assignment to Canvas.

Below is an example question and answer.

Q: Are there any “legitimate” page faults that are not the result of an application error? If so, describe the circumstances under which “legitimate” page faults can occur and what the page fault handler does in each case. If not, why not?

A: There are legitimate page faults, such as page faults encountered when a process attempts to write to a page designated as read-only by a Copy-On-Write implementation (requiring the page fault handler to copy the page and update the appropriate page table entry) or when a page has been swapped out to disk, requiring the page fault handler to identify a victim physical frame to swap out in order to bring the correct physical frame corresponding to the desired virtual page into main memory.

Grading

Each assignment will be graded in three parts. Thirty points will be assigned as follows:

1. You will receive ten points if the work you've submitted indicates a bare minimum of effort (e.g. it’s not copied from a homework for another class or from the textbook).
2. You will receive ten points based on the feedback received from three of your peers.
3. You will receive ten points for providing short, constructive feedback on your peers' answers.

Due Date

This assignment is due on **Wednesday, November 4th by 11:59 PM EDT**. Remember to convert this time to the timezone you currently reside in.

Question 1

Virtual addressing is one of the most innovative ideas in computer science. How does virtual memory keep address spaces separate, and how can it share information between address spaces?

Question 2

What is the *maximum* number of times a single "**`ret`**" instruction could access physical memory? In other words, how many accesses to physical memory could occur before the next instruction is executed? Explain why each of these memory accesses occurs.

Assume that the stack pointer is *not necessarily aligned* when the instruction is executed, so a single memory access to the stack may span multiple cache blocks or even pages. Additionally, assume that *no page faults* occur during the execution of this instruction, so every access has a valid physical mapping. Furthermore, assume the machine conforms to x86-64 architecture with 4 levels of page tables.